



Epasswd

Solving the Heterogeneous *passwd* Program Problem

Eric Davis
Numerical Aerospace Simulation Facility
NASA Ames Research Center

Cracking Passwords

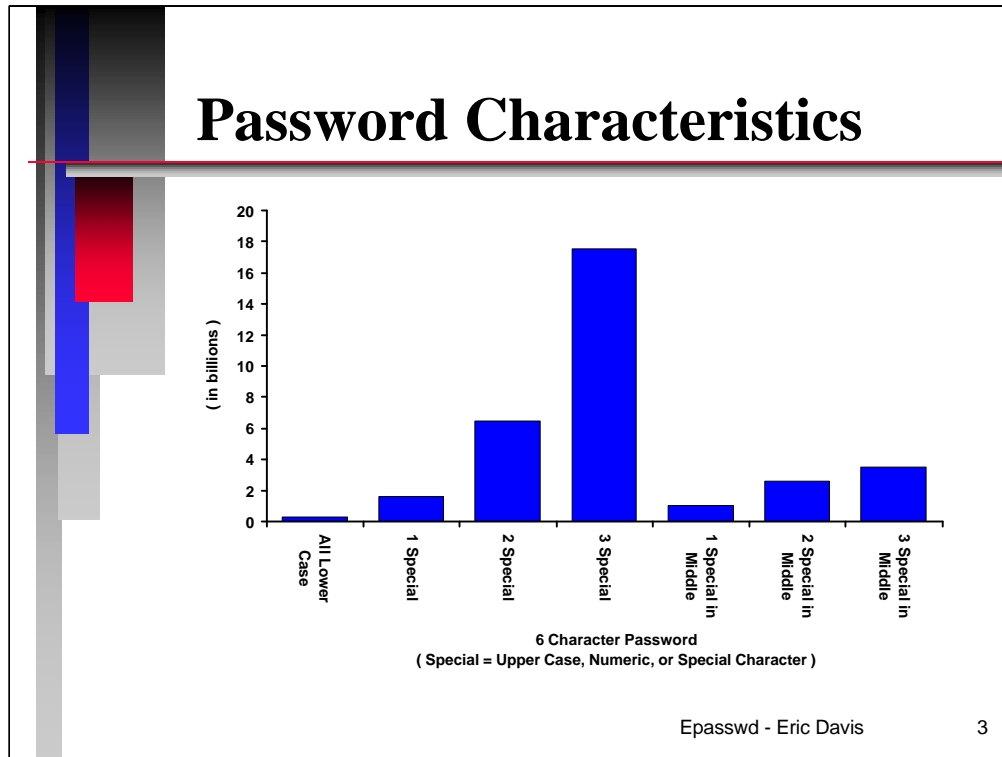
- Alec Muffett's password cracker - "Crack"
- Many dictionaries
- Uses the GECOS field and login name
- Uses various system information
- ANY password derived from ANY dictionary word (or personal info), modified in ANY way, constitutes a potentially guess-able password!

Epasswd - Eric Davis

2

The Security Team ran Crack against the NAS password files and found more than 25% of the passwords were cracked. This was unacceptable because the login password is the only method of user authentication available.

If you can slam the door in the face of potential crackers with a robust password file, you have sealed most of the major avenues of attack immediately. It is much easier for a cracker to break into a system via a cracked password and spoof his/her way to root via operating system holes.



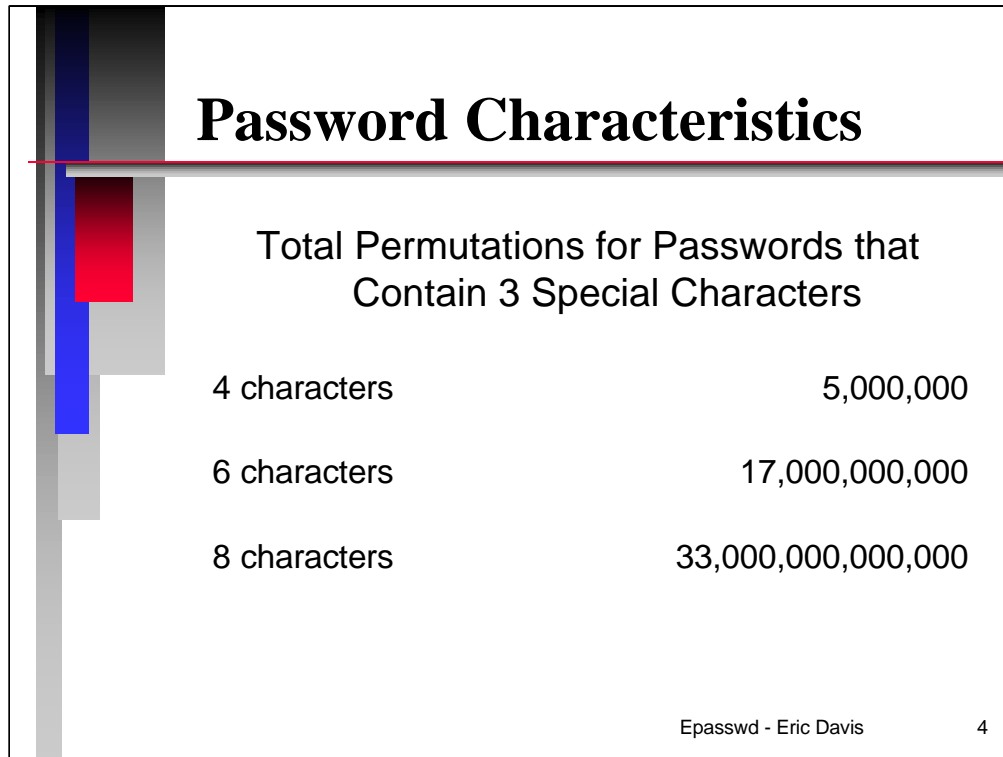
Character sets used by Epasswd

- Lower case characters: [a-z]
- Upper case characters: [A-Z]
- Numeric characters: [0-9]
- Special charcters: [~`!@\$%^&*()_+={}[]\|:;'"<>,.?/<space>]

Example Passwords (special = upper case, numeric, or special character)

- All lower case: bicycle
- 1 special character: Bicycle
- 2 special characters: Bicycle3
- 3 special characters: B1cycle*
- 1 special character in middle: b1cycle
- 2 special characters in middle: b1*cycle
- 3 special characters in middle: b1*cyc7e

Crack has a hard time cracking passwords that contain 2 or more special characters, especially when some exist in the middle of the password.



Here the difference can be seen in the total number of permutations of a password with 3 special characters (special = upper case, numeric, and special characters).

The total number of permutations between a 6 and 8 character password is so significant that it's worth making the user select an 8 character password and remember 2 extra characters.



Password Requirements

- DON'T choose a password that could be found in *any* dictionary
- DON'T choose a password that is similar to the old one
- DON'T choose a password that uses the login name or any variation of it
- DON'T choose a password that contains all lower case or all upper case characters

Epasswd - Eric Davis

5

The dictionaries used by Crack are huge and range from the normal to the bizarre (e.g. Star Trek words dictionary). Crack tests each word in these dictionaries as well as each word written backwards, punctuation at the end, punctuation at the beginning, replacing every 't' with a '3', capitalized, etc. It is possible to configure Crack to use over a thousand different mangling rules for use on each word in the dictionaries.

It is also good to force the user to select a password that differs from the old by at least 3 characters. If an old password gets cracked the cracker will most likely try to login the user's account using the old password with some different characters.



Password Requirements

- DO choose a password that is at least 6 characters long
- DO choose a password that has a mixture of lower case, upper case, numeric, and special characters
- DO choose a completely different password during every change

Epasswd - Eric Davis

6

The Security Team at the NAS requires that all passwords:

- be at least 6 characters long
- contain a lower case character
- contain a upper case character
- contain a numeric character
- contain a special character
- differ from the old password by three characters

Existing Applications

- Default OS
 - not good password checkers
- passwd+
 - good password checker
 - uses a configuration file
 - not well supported
- npasswd
 - good password checker
 - specific for NIS/NIS+
 - very well supported

Epasswd - Eric Davis

7

The *passwd* applications that ship with most UNIX OSs don't force the user to choose strong un-crackable passwords.

Passwd+, written by Matt Bishop, is a good proactive password checker that forces user's to choose good passwords. It uses a run-time configuration file that specifies the construction guidelines when choosing a new password. Unfortunately, it is not well supported.

Npasswd, written by Clyde Hoover, is another good proactive password checker. It offers full support for NIS/NIS+ and is very well supported.

Epasswd - Why?

- Proactive password checker
- Native support for both password aging and shadow passwords
- No extra libraries / system configuration
- Same command line arguments across platforms
- Easy compile time configuration

Epasswd - Eric Davis

8

The existing *passwd* applications did not suit the needs or fulfill the requirements at the NAS. It was decided a new application be developed with the following design criteria:

- Proactive password checking while the user is changing his/her password
- Easy configuration for enforcement of strong un-crackable passwords
- Configuration done at compile time
- Full native support for the underlying OSs password implementation (e.g. password aging, shadow passwords, etc)
- No enhancements/changes would be made to the underlying password implementation which would break other applications and cause re-compiling
- Easily portable to many UNIX platforms

Password Aging

- Specifies how long a password can be used
- Shadow passwords are likely not needed on most platforms
- Minimum age
- Maximum age
- Warning time

Epasswd - Eric Davis

9

If password aging is turned on then *passwd* checks to see if the old password has 'aged' sufficiently. This 'age' specifies the amount of time (in days) that must elapse between password changes. If the password has not aged then the user cannot change it.

A maximum age can also be specified for a password. This age informs the system that the password must be changed after a certain amount of time (in days). After the time has expired, the user is prompted to change the password immediately and cannot log in until the change is made.

A warning age can also be used which tells the system when (in days) to warn the user that his/her password will be expiring soon.

Note, if the password ages are too long, the password can get cracked. If the age is too short, the user might start writing his/her password down. Three months is a good age for a password.

If shadow passwords are being used, the aging information is stored in the */etc/shadow* file, else it is stored in */etc/passwd* within the encrypted password field (a ',' separates the encrypted password and aging info).

Shadow Passwords

- /etc/shadow
- Access restricted (No general read permissions)
- Ability to track account inactivity
- Ability to expire an account

Epasswd - Eric Davis

10

If shadow passwords are turned on, the encrypted passwords are stored in the /etc/shadow file. The encrypted password field in /etc/passwd is '*' out.

The difference between the /etc/passwd and /etc/shadow files is that the shadow file does not have general read permission. In fact, it has the permissions of 0600 (read and write by root only). This eliminates anyone, except root, from grabbing a copy of the password file for cracking.

On some systems, shadow passwords offer the ability to expire accounts immediately, or after a period of inactivity has occurred for the account.

Configuration

- Maximum number of attempts
- Minimum password length
- Maximum password length
- Minimum number of characters in which the new password must differ from the old

The maximum number of attempts specifies how many tries the user gets to enter in a new password that satisfies all the configuration requirements.

The minimum length specifies the minimum number of characters the password must contain. **DON'T MAKE THIS LOWER THAN 6.**

The maximum length specifies the maximum number characters the password can contain. On most UNIX systems the *crypt* system call uses only the first 8 characters of the password.

Configuration

- Minimum number of lower case
- Minimum number of upper case
- Minimum number of numeric
- Minimum number of special
- Minimum number of tests from above that a password must pass
- Specify that an upper case, numeric, or special character must exist in the middle of the password

Epasswd - Eric Davis

12

There are four main tests that can be configured: minimum lower case, minimum upper case, minimum number, and minimum special characters. The minimum number of these tests that must be passed is specified.

Lastly, a user can be forced to have an upper case, numeric, or special character in the middle of the password. Not in the first or last character position.

For example with the following configuration:

- minimum lower case = 1
- minimum upper case = 1
- minimum numeric = 1
- minimum special = 1
- minimum tests pass = 3
- upper case middle = false
- numeric middle = true (if a numeric exists, it must be in the middle)
- special middle = true (if a special exists, it must be in the middle)

These passwords are valid: Bic1cle, b!c1cle, Bic*cycle, bicYc%le

These passwords are invalid: bicycle, Bicycle!, BicYcle, 1bicyclE

Command Line Arguments

- (-s) list information for account
- (-a) list information for all accounts
- (-i) set the number of days in which the account can be inactive before expiring
- (-e) set the date in which the account expires
- (-E) expire the account
- (-c) insert a new encrypted password from the command line

Epasswd - Eric Davis

13

The '-s' option lists the attributes for an account. This includes the login name, uid, gid, home directory, password status, etc. The password status can be: NP for no password, LK for locked, and PS for normal password.

If shadow passwords are turned on, the system administrator can use the '-i', '-e', and '-E' options to expire a password.

The '-c' option is used to swap in a new encrypted password from the command line. For example:

```
% passwd -c q4mjzTnu8ic/F edavis
```

This causes the encrypted string to be swapped into the password field for edavis. The encrypted string must be exactly 13 characters long and be created from the following set of characters: [a-zA-A0-9./].

Command Line Arguments

- Password aging
 - (-n) minimum age
 - (-x) maximum age
 - (-w) warn time
 - (-f) expire password and force change
- (-l) lock password
- (-h) prints the usage
- (-v) prints system information and the number of days since Jan 1,1970

Epasswd - Eric Davis

14

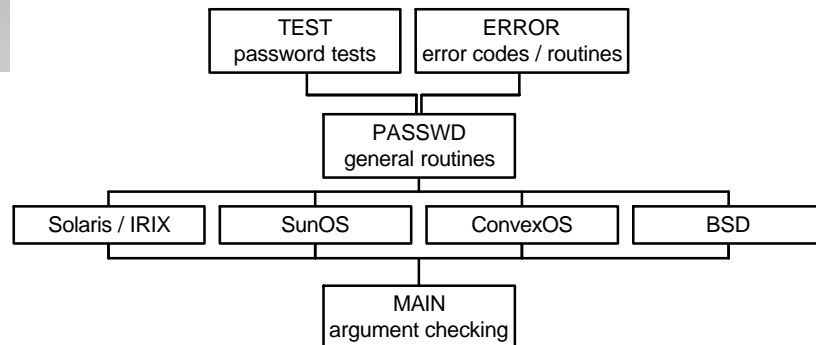
The command line arguments for password aging allow the system administrator to set the minimum, maximum, and warning ages for a user's password. The administrator can also expire a password which forces the user to change his/her password the next time he/she logs in.

When a password gets locked a '*' is swapped into the encrypted password field. This causes successive logins to fail.

The '-v' option causes Epasswd to print system specific information such as the uname, whether or not shadow passwords are turned on, and the number of days since Jan 1, 1970. The number of days is useful because password aging information is specified in days since Jan 1, 1970.

Porting Epasswd

Object Hierarchy



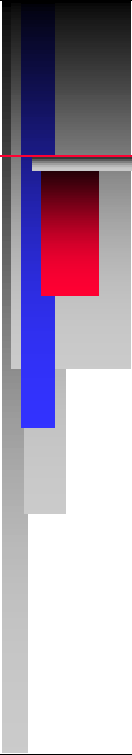
Epasswd - Eric Davis

15

Epasswd was written in C++ and designed to allow porting across platforms as easy as possible. This modularity allows the programmer to easily plug in object definitions for each platform.

Object Design and Definitions

- TEST: contains all the functions used for testing passwords
- ERROR: contains all the error functions and return codes
- PASSWD: generic functions used across all platforms
- OS SPECIFICS: platform specific functions (object named after platform)
- MAIN: argument checking



Epasswd - Current

- In production at the NAS since March 97
- No passwords created using Epasswd have been cracked
- NASA UNIX Expert Center - recommends Epasswd as a default UNIX configuration

Epasswd - Eric Davis

16

The NAS has been using Epasswd for over a year. The Security Team runs Crack on a regular basis and all new passwords which have been created using Epasswd have never been cracked.

The NASA UNIX Expert Center (at the Lewis Research Center in Cleveland, Ohio) has given full endorsement of Epasswd and recommends that Epasswd be installed on all UNIX systems within NASA.

Epasswd - Future

- As many ports as possible!
- Add hooks into Cracklib
- Logging to a log file
- Ability to store old passwords to prevent users from using them again
- Add to the Mod-Auth Apache module

Epasswd - Eric Davis

17

Cracklib is a UNIX library, written by Alec Muffett, which contains simple routines which try to crack encrypted passwords using some basic mangling rules. Note, it is highly doubtful that Cracklib could immediately crack any password created by Epasswd.

Epasswd should log all activity to a log file for tracking.

It would also be nice to store all old encrypted passwords in a file to prevent users from using them again. This eliminates the possibility of when a password expires, the user won't change the password and then quickly change it back to the old one.

Adding to the Mod-Auth Apache module could allow users to change their web-site passwords remotely. This was an idea that came up and has not been thought out yet.

URLs

- Epasswd
 - <http://www.nas.nasa.gov/~edavis/epasswd/>
- Passwd+
 - <ftp://ftp.dartmouth.edu/pub/security/>
- Npasswd
 - <http://uts.cc.utexas.edu/~clyde/npasswd/>
- Crack
 - <http://www.users.dircon.co.uk/~crypto/>